



Publication number : **0 574 199 A1**

**EUROPEAN PATENT APPLICATION**

Application number : **93304367.1**

Int. Cl.<sup>5</sup> : **G06K 15/00, G06K 15/10**

Date of filing : **04.06.93**

Priority : **10.06.92 US 896367**

Date of publication of application :  
**15.12.93 Bulletin 93/50**

Designated Contracting States :  
**DE FR GB IT**

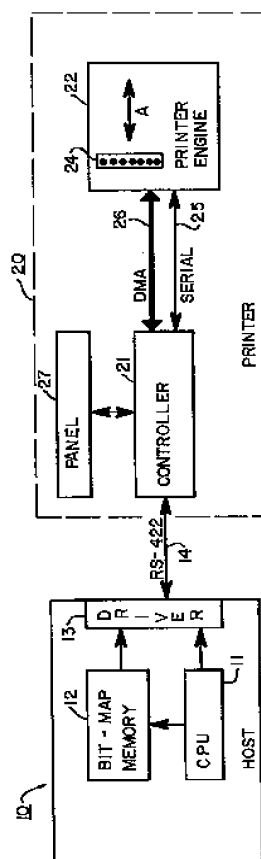
Applicant : **CANON INFORMATION SYSTEMS, INC.**  
**3188 Pullman Street**  
**Costa Mesa, CA 92626 (US)**

Inventor : **Russell, William C., c/o Canon Info. Systems, Inc.**  
**3188 Pullman Street**  
**Costa Mesa, CA 92626 (US)**  
Inventor : **Emerson, H. Brad, c/o Canon Info. Systems, Inc.**  
**3188 Pullman Street**  
**Costa Mesa, CA 92626 (US)**  
Inventor : **Ip, Tony K., c/o Canon Info. Systems, Inc.**  
**3188 Pullman Street**  
**Costa Mesa, CA 92626 (US)**

Representative : **Beresford, Kelth Denis Lewis et al**  
**BERESFORD & Co. 2-5 Warwick Court High Holborn**  
**London WC1R 5DJ (GB)**

**Method and apparatus for image data conversion in a printer.**

Method and apparatus for converting image data from row format into column format and a printer incorporating the same. A conversion circuit, which is responsive to writes to a first address space, is provided for converting the image data from the row format into the column format. A CPU writes the row format image data to the first address space whereupon the conversion circuit converts it to the second format. The CPU then writes image data to a second address space. In accordance with an unwind flag set by the CPU, either the converted image data or the original image data is stored in a memory. After the data has been written to memory, the CPU initiates a DMA transfer of the image data to a printer engine.



**FIG. 1**

## BACKGROUND

### Field Of The Invention

The present invention pertains to a method and apparatus for reformatting image data prior to printing the data on a printer, and in particular to a method and apparatus for reformatting from a row format to a column format in situations where the image data is sent to the printer row-by-row but where the printer prints the image data column-by-column.

### Description Of The Related Art

Printing apparatuses are widely used in connection with data processing, office automation and personal computer equipment. With advances in technology, it is now possible for printers to have a print head that includes many print elements that are closely spaced with respect to each other. For example, one type of printer currently available includes a single print head having 64 bubble jet nozzles arranged in a single column. Such an arrangement permits speedier printing because an entire "band" of print information can be printed with a single sweep of the print head across the printer carriage.

Image data to be printed by such a printer is ordinarily stored in a bit map memory, that is, each pixel of image data is represented by a separate bit in a byte addressable memory. Due to the above-described structure of the print head, it is advantageous to organize the bit map memory in columns of image data. This organization is advantageous because it allows the print head to print pixels of image data in the same order as bits of image data are read out from the bit map memory.

Figure 8(a) depicts the desired column-ordered organization for the bit map memory of a band of image data for a typical printer which includes a print head having 64 bubble jet nozzles arranged in a vertical column. The carriage width for the printer is 15 inches across which is printed at 360 dots per inch ("dpi"). Accordingly, as shown in Figure 8(a), each band of bit map image data is 5400 bits across and 64 bits high. The individual bits in the bit map memory are sequentially ordered in columns. That is, the first 64 bits of bit map memory, namely bits 0 through 63, correspond to the first column of pixels printed by the print head, the second 64 bits of bit map memory, namely bits 64 through 127, correspond to the second column of pixels and so on through the last 64 bits of bit map memory, namely bits 345536 through 345599, which correspond to the 5400th column of pixels printed by the print head.

Typically, the image data in a bit map memory is generated by a host CPU which executes an application program, and the image data is generated and stored in the bit map memory in accordance with the

application program. In contrast to the column organization shown in Figure 8(a), most application programs typically generate image data in row order and store the image data in rows. Figure 8(b) shows this row-ordered organization for the same band of image data shown in Figure 8(a). The first row includes bits 0 through 5399, the second row contains bits 5400 through 10799, and so on through the 64th row which contains bits 340200 through 345599.

In the case where the application program in the host computer generates and stores image data in the order shown in Figure 8(b), it is not possible for a printer of the type in question to print the image data because the individual bits are read from memory in the wrong order. Nor is it practical to skip through a row-ordered-bit map memory so as to read out the image data in column order. This is because the bits in bit map memory are grouped into eight bit units called "bytes". For example, referring either to Figure 8(a) or Figure 8(b), the first byte contains bit 0 through bit 7, the second byte contains bit 8 through 15, and so on through the 43200th byte. This byte organization makes it difficult to access individual bits within the bytes, and in particular makes it difficult to read, for example, first bit 0 from the first byte in Fig. 8(b), then bit 5400 from the 676th byte, and so on.

To reformat the bit map image data from the Figure 8(b) row-ordered format to the Figure 8(a) column-ordered format, it has been considered to program the host CPU to extract desired bits from the Figure 8(b) bit map memory and to construct a bit map memory in accordance with Figure 8(a). For several reasons, such an arrangement has proven unsatisfactory. First, because of the byte organization of the bits in the bit map memory, the bit-wise extraction and construction of bit map image memories is a time consuming and CPU-intensive process. The host CPU is well-suited for operating on bytes, but it is not well-suited for operating on individual bits within the byte. Thus, for example, to extract an individual bit from a byte requires the CPU to perform shift operations, masking operations and Boolean logic operations. Similarly, to add a bit to a byte requires Boolean logic operations. Each of these operations takes a finite amount of time and when multiplied by the number of bits in a band (345,600 in the case of Figure 8(b)) results in unacceptably slow operation. This is especially true in the case of halftone or color image data where several bit map planes may be required.

Second, the primary task for the host CPU is to execute the application program. Any other tasks such as bit map conversions detract from the time available for the host CPU to execute the application program and unacceptably reduce performance.

Third, even if bit map conversion could be accomplished in an acceptably short period of time, the conversion steps required in one type of microprocessor, e.g., an Intel 80386, might not necessarily be the

same as those required in a second type of microprocessor, e.g., a Motorola 68000. This is because bit operations are at the most fundamental level of a microprocessor. Accordingly, bit operations for one type of microprocessor are different from bit operations of another type of microprocessor, and a different set of print instruction steps would be needed for each individual type of microprocessor.

It has also been considered to provide a microprocessor in the printer's controller for performing the Figure 8(b) to Figure 8(a) conversion. This has also proven to be unsatisfactory because it simply transfers the above noted problems from the host CPU to the printer controller's CPU. Moreover, this approach requires the controller to have a highly sophisticated microprocessor which is inconsistent with the low cost objective for the controller.

### SUMMARY OF THE INVENTION

The present invention solves the above problems by providing a method and apparatus for reformatting (or unwinding) graphics images in a bit map memory from a row-ordered format to a column-ordered format.

In one aspect, the invention is an apparatus for reformatting image data from a first format into a second format comprising converting means responsive to image data written to a first address space for converting the image data from the first format to the second format, selecting means for selecting the first format or the second format, and output means responsive to image data written to a second address space for outputting either the image data written to the second address space or the image data converted by said converting means in accordance with the selection by said selecting means.

In another aspect, the invention is an apparatus for reformatting image data from a first format into a second format comprising writing means for writing the image data to a second address space, converting means responsive to said writing means writing image data to the first address space, said converting means for converting the image data from the first format into the second format, and outputting means responsive to said writing means writing to the second address space, said outputting means for outputting data converted by said converting means.

In another aspect, the invention is a printer comprising a serial interface, a printer engine including a print head adapted to print plural rows of print information simultaneously, and a controller for controlling said printer engine in accordance with image data and commands received on said serial interface, said controller including conversion means responsive to image data written to a first address space for converting the image data from a first format into a second format, and output means for outputting the convert-

ed image data to said printer engine, said output means being responsive to image data written to a second address space.

In another aspect, the invention is a printer comprising a serial interface adapted to receive serial image data, a printer engine including a print head adapted to print plural rows of image data simultaneously, a first memory for storing process steps that select a first or a second format, that write image data to a first address space and to a second address space, and that initiate a direct memory transfer, or DMA transfer, from the second address space to said printer engine, process means for executing the process steps stored in said first memory, converting means responsive to said process means writing to the first address space for converting image data from the first format to the second format, a second memory responsive to said process means writing to the second address space for storing image data, selection means for selecting the first format or the second format in accordance with the process step executed by said process means, said selection means for causing the second memory to store converted image data from said conversion means or unconverted image data in accordance with the selected format, and a DMA channel for initiating DMA transfer from said second memory to said printer engine.

In another aspect, the invention is a method for converting image data from a first format to a second format comprising the steps of writing first format image data to a first address space, converting first format image data written to the first address space into second format image data, writing to a second address space, and storing data converted in said converting step in response to writing to the second address space.

In another aspect, the invention is a method for printing image data comprising the steps of receiving image data and commands over a serial interface, writing the image data to a first address space, converting the image data written to the first address space from a first format into a second format, writing to a second address space, storing image data converted in said converting step in response to writing to the second address space, and printing the image data stored in said storing step.

### BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram showing a host CPU and a printer;

Figure 2 is a block diagram showing a controller for the Figure 1 printer;

Figure 3 is a diagram for explaining memory organization in the Figure 2 controller;

Figures 4, comprising Figures 4(a), 4(b) and 4(c) are detailed block diagrams showing the unwinding logic of the Figure 2 controller;

Figure 5 is a flow diagram to illustrate how image data is selected for unwinding;

Figure 6(a) is a chart illustrating pixel layout, Figure 6(b) is a diagram of row-ordered bit map image data, and Figure 6(c) is a diagram of column-ordered bit map image data;

Figure 7 is a flow chart for the Figure 2 controller; and

Figure 8, comprising Figures 8(a) and 8(b) depict the arrangement of bits in a column-ordered bit map memory and a row-ordered bit map memory, respectively.

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 depicts an overall block diagram of a host computer 10 interconnected with a printer 20. As shown in Figure 1, host computer 10 includes CPU 11, bit map memory 12 and printer driver 13. As is known in the art, CPU 11 executes an application program which includes steps for forming a bit map image in memory 12.

To print the bit map image so formed, CPU 11 initiates a call to printer driver 13. The printer driver transmits commands to printer 20 over interface 14, for example, to configure the printer to accept a particular type of data or to interrogate the printer to determine the printer's condition. The printer driver then sequentially accesses each byte in bit map memory 12 and transmits the data over interface 14, such as an RS-422 serial interface, to printer 20.

The image data transmitted by the printer driver may be in compressed or uncompressed format. That is, rather than transmitting each and every byte in bit map memory 12, the printer driver may compress the data, for example through run-length or Huffman encoding, and transmit the compressed data so as to shorten the transmission time.

Printer 20 includes controller 21 which controls operation of printer engine 22 that includes a print head 24 having plural print elements arranged in a vertical column. Print head 24 is arranged in printer engine 22 to reciprocate across the printer's carriage in the direction indicated by double headed arrow A. Controller 21 communicates with printer engine 22 through a serial interface 25 so as to send and to receive commands to and from the printer engine. Actual print data, however, is transmitted from controller 21 to printer engine 22 through a direct memory access ("DMA") channel 26. Such an arrangement, though not necessary, is preferred since it frees controller 21 from overseeing the transfer of image data to the printer engine.

Controller 21 also interfaces with panel 27 located in the printer housing. The panel includes a number of control push buttons, a speaker, and a series of indicator lamps by which the operator can com-

mand specific operations (for example, form feed and on/off line) and by which the operator can monitor the status of the printer.

In response to the mixed transmission of image data and commands over interface 14, controller 21 receives the mixed transmission and separates commands from image data. The commands are executed. As for the data, if it is compressed, the controller uncompresses it and stores it in a bit map memory in the same order as bit map memory 12 in the host CPU. As explained more fully below, the controller then unwinds the image data from a row-ordered format into a column-ordered format and stores the unwound bit map image data for DMA transfer over DMA interface 26 to printer engine 22. The DMA transfer is coordinated over serial interface 25 in accordance with well-known techniques.

Figure 2 is a detailed block diagram of controller 21. As shown in Figure 2, controller 21 includes a CPU 30 such as an NECV25 microprocessor. Controller 21 further includes interface connector 31 for connecting to the interface 14, interface logic 32 for converting the data on the interface into parallel data (if necessary) for use by the CPU 30, erasable programmable read only memory ("EPROM") 34 for storing process steps for execution by CPU 30, a speaker 35 and a crystal 36 for controlling the cycle time of CPU 30. Controller 21 further includes a dynamic random access memory ("DRAM") 37 and DRAM data control 39 for controlling access to DRAM 37. DRAM 37 is for storing uncompressed bit map image data received over interface 14 and also for storing commands received over interface 14.

Controller 21 further includes unwind data logic 40 for converting a row format bit map image data into column format bit map image data, unwound data buffer 41 for temporarily storing the unwound data, and static random access memory ("SRAM") 42 for storing column format bit map image data. It is preferable for SRAM 42 to be double buffered as depicted at 42a so as to permit construction of column format bit map image data simultaneously with DMA transfer of previously constructed bit map image data. Double buffering is controlled by programmable array logic ("PAL") 38, as more fully described below.

Actual DMA transfer to the printer engine is accomplished from printer data control 44 and printer command logic 45, both of which interface with printer engine interface connector 46. Connector 46 includes the aforementioned serial interface 25 for controlling the interface between the printer engine and the controller, and DMA channel 26 which permits printer engine 22 to access SRAM 42 directly.

Address bus 47 and data bus 49 are provided for routing data among the various components in the controller.

Memory organization in controller 21 is shown in Figure 3 which gives correspondence between a

memory address and a physical memory device. As shown in Figure 3, memory addresses from 00000h (wherein "h" designates a hexadecimal number) through 3FFFFh correspond to DRAM 37. Memory addresses from 80000h through 8FFFFh do not correspond to any physical memory device; rather, when data is written to one of these addresses the data is directed to unwind data logic 40. Memory addresses from D0000h through EFFFFh refer to addresses in double-buffered SRAM 42 and 42a, wherein the addresses from D0000h to DFFFFh refer to the first buffer 42 and addresses from E0000h to EFFFFh refer to the second buffer 42a. Memory addresses from F0000h through FFFFFh correspond to EPROM 34. Activation of the particular device (i.e., EPROM 34, DRAM 37, unwind data logic 40 and SRAM 42) together with any desired double buffering is controlled by chip select signals from PAL 38 as described more fully below.

Figure 4, comprising Figures 4(a), 4(b) and 4(c), provide a detailed block diagram of unwind data logic 40 as it interacts with other components in controller 21. As shown in Figure 4(a) PAL 38 is provided with the four most significant address bits on address bus 47, as well as a read/write signal "R/W" from CPU 30, a memory strobe signal "MSTB" from CPU 30 and a memory required signal "MREQ" from CPU 30. Based on these signals, PAL 38 generates an unwind logic select signal when an address from 80000h to 8FFFFh appears on address bus 47, an SRAM chip select signal (SRAMCS) when an address from D0000h to EFFFFh appears on address bus 47, a DRAM chip select signal (DRAMCS) when an address from 0000h to 3FFFFh appears on address bus 47, an EPROM chip select signal (EPROMCS) when an address from F0000h to FFFFFh appears on address bus 47, and two chip select signals for the SRAM (SRAMCS1 and SRAMCS2) that coordinate double buffering for the SRAM in accordance with whether an address from D0000h to DFFFFh or from E0000h to EFFFFh appears on the address bus.

A pair of OR gates 50 and 51 are connected to the R/W, MSTB, and MREQ outputs of CPU 30 and generate a memory write signal (MEMWR).

In accordance with commands transmitted from driver 13 on interface 14, CPU 30 outputs an unwind flag (UNWIND) to permit de-selection of the unwind logic in the event that data transmitted by host computer 10 is already in the appropriate order. For example, in the case of landscape images, unwind logic may not be needed and the unwind flag is reset to zero.

Buffer 52 is provided to transfer each of image data bits d0 through d7 on data bus 49 directly to SRAM 42 via BMD bus 56 in the event that unwinding is not desired. Buffer 52 is enabled by OR gates 54 and 55 when an SRAM address appears on the address bus (as indicated by the SRAMCS signal from-

PAL 38), when the memory write signal appears, and when the unwind flag is low. When this combination of events occurs, buffer 52 is enabled and transfers the data on data bus 49 to BMD bus 56. The BMD bus 56, in coordination with double buffering as controlled by PAL 38, feeds data to SRAM 42 (or 42a) for subsequent DMA transfer to the printer engine (see Figure 2).

To print image data in the event that unwinding is not desired, CPU 30 sets the unwind flag low. Then each byte of bit map memory is retrieved from DRAM 37 and written to SRAM 42. By writing to SRAM 42 with the unwind flag low, CPU 30 circumvents unwind logic 40 and causes the image data to be transferred directly to BMD bus 56 and stored directly in SRAM 42 for subsequent DMA transfer to the printer engine.

Buffer 57 is provided to transfer image data on data bus 49 to unwinding logic 40 in the event that unwinding is desired. With the unwind flag high (as set by CPU 30), buffer 57 is enabled when an address from 80000h to 8FFFFh appears on the address bus, as indicated by the unwind logic select signal from PAL 38, and transfers data bits d0 through d7 on data bus 49 to BD bus 59 for subsequent unwind processing.

As shown in Figure 4(b), the BD bus is connected to each of eight 8-bit latches 60-67. The clock inputs for latches 60-67 are provided from monostable multivibrators 69a and 69b, the inputs for which are provided by multiplexer 68. Multiplexer 68 is enabled when an address from 80000h to 8FFFFh appears on address bus 47, as indicated by the unwind logic select signal from PAL 38, and when the memory write signal MEMWR is generated by NAND gate 5. When enabled, multiplexer 68 decodes the three least significant binary digits on the address bus (i.e., bits A0, A1 and A2) to select one of outputs y0 through y7. The selected output activates monostable multivibrator 69a or 69b which generates the clock signal for the selected corresponding latch.

In operation, CPU 30 retrieves a byte of row-ordered image data from the bit map image data in DRAM 37 and writes the row format byte to an address in the range from 80000h to 8FFFFh. PAL 38 detects the presence of this address on the address bus and generates the unwind logic select signal. The unwind logic select signal enables both buffer 57 and multiplexer 68. Buffer 57 transfers the image data byte to BD bus 59 and multiplexer 68 latches the transferred byte in accordance with the three least significant bits on the address bus 47. Eight bytes of image data from bit map memory in DRAM 37 are sequentially written in this manner to load each of latches 60-67 and provide 64 total bits of image information. The 64 bits of image information are provided to the LB bus 70 which, in turn, is connected to the inputs of buffers 71 through 78 shown in Figure 4(c).

As shown in Figure 4(c), each of buffers 71 through 78 includes eight inputs which are connected

to corresponding ones of the outputs from latches 60 through 67. Thus, for example, the eight inputs for buffer 71 are respectively connected to bit LD1 from latch 60, bit LD9 from latch 61, bit LD17 from latch 62, etc. Through this interconnection, the row-ordered bytes of image data stored in latches 60 through 67 are converted to column-ordered bytes of image data.

Buffers 71 through 78 are enabled by the outputs of multiplexer 79. Multiplexer 79 is similar to multiplexer 68, but unlike multiplexer 68 multiplexer 79 decodes the next three least significant bits on address bus 47 (i.e., bits A3, A4 and A5). In addition, unlike multiplexer 68 which is enabled by a write to an address in the range 80000h to 8FFFFh, multiplexer 79 is enabled by writes to SRAM 42 or 42a, as indicated by the SRAMCS signal from PAL 38.

In operation, once eight bytes of image data have been latched into latches 60-67 (and, consequently, converted to column-format image data by buffers 71-78), CPU 30 writes eight bytes of data to SRAM 42 or 42a. Because the unwind flag is high, buffer 52 ignores the data written by CPU 30 during these writes and whatever data is written by the CPU to data bus 49 is not transferred to BMD bus 56. Rather, multiplexer 79 is enabled by the SRAMCS signal and sequentially activates buffers 71-78 in accordance with bits A3, A4 and A5 on address bus 47.

As the buffers are actuated, the eight bits unwound by each buffer are placed on MD bus 80. The eight unwound data bits on MD bus 80 are buffered to BMD bus 56 by buffer 81 which is enabled in accordance with the SRAMCS signal, the memory write signal and the unwind flag, as processed by OR gates 82 and 83, respectively. Because the unwind flag is high, unwound, or column-ordered, data is placed on the BMD bus when CPU 30 writes to SRAM 42 or 42a. As before, the BMD bus, in coordination with double buffering controlled by PAL 38, feeds image data to SRAM 42 (or 42a) for subsequent DMA transfer to the printer engine.

In sum, to unwind data from row-ordered format to column-ordered format, CPU 30 first sets the unwind flag. Then, CPU 30 retrieves eight bytes of row-ordered data from bit map memory in DRAM 37 and writes the eight bytes to an address between 80000h and 8FFFFh. The eight bytes are latched into latches 60 through 67. CPU 30 then writes the eight bytes to an SRAM address whereupon buffers 71 through 78 place column-ordered data first onto the MD bus 80 and then onto the BMD bus 56. Data on the BMD bus 56 is channeled to SRAM 42 or 42a in coordination with double buffering from PAL 38. The double buffered data in SRAM 42 is then DMA transferred to the printer engine under the control of printer control 44 and printer command logic 45.

Figure 6, comprising Figures 6(a), 6(b), and 6(c), are views for describing how CPU 30 selects specific bytes of row-ordered image data from DRAM 37 and

unwinds them into column-ordered image data in SRAM 42.

Figure 6(a) identifies each and every pixel in a band of image data suitable for the print head in question and provides a unique label for each pixel. The pixels are shown in their desired physical location on a print medium or other visualization medium. As shown in Figure 6(a), pixels in the first row are all labelled "AA" followed by the column number (that is, AA0 through AA5399). Pixels in the second row are all labelled "AB" followed by the column number (that is, AB0 through AB5399), and so on until pixels in the 64th row which are all labelled "HH" followed by the column number (that is, HH0 through HH5399).

Figure 6(b) shows how the pixels of Figure 6(a) are stored in row-ordered bit map memory in DRAM 37. In Figure 6(b), it is assumed that driver 13 has transmitted image data from bit map memory 12 in a row-by-row order. Other orders are possible and CPU 30 can be programmed through commands from the driver to respond to other orders. In particular, it is possible for driver 13 to send a command over interface 14 to CPU 30 so that CPU 30 expects data to be transmitted in a different order.

As shown in Figure 6(b), in row-ordered bit map memory, pixels are stored in DRAM 37 sequentially by rows. Thus, the first byte (address 0h) in bit map memory contains the first eight pixels of row 1, namely bits AA0 through AA7. The second byte (address 1h) continues row 1 and contains bits AA8 through AA15 and so on through the end of row 1, corresponding to the 675th byte (address 2A2h) which contains bits AA5392 through AA5399.

The second row in row-ordered bit map memory begins at the 676th byte in DRAM (corresponding to address 2A3h) and contains bits AB0 through AB7. This sequence continues until the 64th row in bit map memory which begins at the 42526th byte in bit map memory (corresponding to address A61Dh) which contains bits HH0 through HH7, and ends at the 43200th byte in bit map memory (corresponding to address A8BFh) which contains bits HH5392 through HH5399.

Figure 6(c) shows the column-ordered bit map memory stored in SRAM 42 which results after unwind-processing the row-ordered data in DRAM 37, and which is suitable for printing by the print engine shown in Figure 2. Figure 6(c) depicts the addresses corresponding to the first buffer in double-buffered SRAM, that is, addresses from D0000h through DFFFFh. It should be understood that Figure 6(c) is equally applicable to SRAM 42a, that is, the other buffer of the double-buffered SRAM, in which case addresses range from E0000h through EFFFFh.

As shown in Figure 6(c), in column-ordered bit map memory, pixels are stored in SRAM sequentially by columns. Thus, the first byte in bit map memory (address D0000h) contains the first eight pixels of col-

umn 1, namely bits AAO through AH0. The second through seventh bytes (corresponding to addresses D0001h through D0007h) complete column 1 and contain bits BA0 through HH0.

The second column in column-ordered bit map memory begins at the eighth byte of SRAM (corresponding to address D0008h) and contains bits AA1 through AH1. The second column continues with the ninth through fifteenth bytes in SRAM (corresponding to addresses D0009h through D000Fh) and contains bits BA1 through HH1.

The third through 5400th columns are likewise arranged in column order beginning with the 16th location in SRAM (corresponding to address D0010h), continuing through the beginning of the 5400th column (corresponding to address DA8B8h) which contains bits AA5399 through AH5399, and ending at the 43200th byte of SRAM memory (corresponding to address DA8BFh) which contains bits HA5399 through HH5399.

To reformat the row-ordered bit map memory shown in Figure 6(b) into the column-ordered bit map memory shown in Figure 6(c), CPU 30 processes eight-by-eight bit blocks of image data one at a time until the entire band of image data has been processed. In the embodiment described here, CPU 30 processes eight-by-eight bit blocks of data proceeding from the left side of the band to the right side, but this should not be considered limiting. The arrangement is, however, preferable, especially in an embodiment where the SRAM is not double-buffered, since it allows print head 24 to begin left-to-right printing before a complete band of image data has been processed. If this advantage is desired, image unwind processing direction should proceed in correspondence with the printing direction, for example, right-to-left, or alternating directions.

For each eight-by-eight bit block of image data, CPU 30 accesses each of the eight bytes of image data in DRAM 37 that constitute the rows in the block. Most conveniently, the rows are accessed in row order, that is, the first row is accessed first and the eighth row is accessed last, but this is not essential.

For the band of image data shown in Figure 6(a), CPU 30 begins unwinding with DRAM image data corresponding to eight-by-eight bit block 87 which is the upper left-most eight-by-eight bit block of image data. CPU 30 accesses the first row in block 87 which is found in the first byte in DRAM 37 (corresponding to address 0h) and contains bits AA0 through AA7. CPU 30 writes this byte to address 80000h which is obtained by adding 80000h to the DRAM address. As described above, PAL 38 detects that data has been written to this address and sets the unwind logic select signal. Accordingly, the data so written is latched into one of latches 60 through 67 in accordance with the three least significant bits on address bus 47 as decoded by multiplexer 68. In the particular instance

of address 80000h, bits AA0 through AA7 are latched into latch 60.

CPU 30 then accesses the second row of block 87 which is found in the 676th location in DRAM 37 (corresponding to address 2A3h) and which contains bits AB0 through AB7. Again, 80000h is added to this address and the data from DRAM 37 is written to location 802A3h. The three least significant bits of this address are decoded by multiplexer 68 so as to cause bits AB0 through AB7 to be latched into latch 61.

The process is continued for remaining rows 3 through 8 in block 87 (corresponding to DRAM addresses 546h, 7E9h, ABCh, D2Fh, FD1h and 1275h) thereby causing corresponding bits to be latched into latches 62 through 67. In particular, bits AC0 through AC7 are latched into latch 62, bits AD0 through AD7 are latched into latch 63, bits AE0 through AE7 are latched into latch 64, bits AF0 through AF7 are latched into latch 65, bits AG0 through AG7 are latched into latch 66, and bits AH0 through AH7 are latched into latch 67.

By virtue of the above-described interconnection of latches 60 through 67 to buffers 71 through 78, unwound, column-ordered image data is ready to be transferred from buffers 71 through 78 into SRAM. Specifically, because of the interconnection, buffer 71 stores bits AA0 through AH0: bit AA0 from latch 60, bit AB0 from latch 61, bit AC0 from latch 62, and so on. Likewise, buffer 72 stores bits AA1 through AH1, buffer 73 stores bits AA2 through AH2, buffer 74 stores bits AA3 through AH3, buffer 75 stores bits AA4 through AH4, buffer 76 stores bits AA5 through AH5, buffer 77 stores bits AA6 through AH6, and buffer 78 stores bits AA7 through AH7. CPU 30 transfers bits from these buffers 71 to 78 by writing a data word (any data word) to the appropriate SRAM address. As described above, if the unwind flag is set, the actual data appearing on data bus 49 is ignored; rather, data appearing on MD bus 80 is transmitted to SRAM 42 via BMD bus 56.

Assuming that the eight-by-eight bit block of image data consisting of bits AA0 through AH7 have been latched into latches 60 through 67 and unwound onto buffers 71 to 78, as described above, CPU 30 writes first to SRAM address D0000h. Bits A3, A4 and A5 on the address bus are decoded by multiplexer 79 to activate buffer 71 and to cause bits AA0 through AH0, which correspond to the first column of block 87, to be stored in SRAM at location D0000h. CPU 30 next writes to SRAM address D0008h. Again, multiplexer 79 decodes bits A3, A4 and A5 on the address bus to activate buffer 72 and to cause bits AA1 through AH1, corresponding to the second column in block 87, to be stored in SRAM at location D0008h. Likewise, CPU 30 transfers the bits from buffers 73 to 78 by writing to SRAM at addresses D0010h, D0018h, D0020h, D0028h, D0030h and D0038h thereby causing the remaining bits (that is, bits AA2

through AH7) to be stored at those addresses in SRAM.

With the first eight-by-eight block of image data transformed into column-ordered format, CPU 30 selects the next eight-by-eight block of image data which in this case is block 88 in Figure 6(a) corresponding to rows 9 through 16 and columns 1 through 8. As before, CPU 30 accesses DRAM corresponding to the rows in block 88, adds 80000h to the addresses accessed, and writes the words so as to latch them into latches 60 through 67. In this case, bits BA0 through BH7, from block 88 are latched into latches 60 through 67. CPU 30 then writes to addresses D0001h, D0009h, D0011h, D0019h, D0021h, D0029h, D0031h and D0039h so as to cause these bits to be transferred from buffers 71 to 78, in column-ordered format, and stored in SRAM as continuations of columns 1 through 8, respectively.

The process proceeds with the next lower eight-by-eight bit block of image data until the first eight columns have been transformed from row-ordered format in DRAM 37 into column-ordered format in SRAM 42. At this point, processing reverts to the next right-most eight-by-eight bit block which corresponds to the first eight rows of image data and the second eight columns of image data in Figure 6(a). In DRAM 37, the image data is at DRAM addresses 0001h, 02A4h, etc., corresponding to bits AA8 through AH16.

Figure 7 is a flow chart showing how eight-by-eight bit blocks of data are selected from the band of image data, and how these bits are unwound from row-ordered format in DRAM to column-ordered format in SRAM. The process steps depicted in Figure 5 are stored in EPROM 34 and executed in CPU 30. As shown in step S501, CPU 30 first sets a DRAM address table consisting of the eight addresses corresponding to block 87 of image data. In step S502, CPU 30 sets a similar address table for SRAM. In step S503, CPU 30 reads row-ordered image data from DRAM 37 in accordance with the first address shown in the DRAM table. 80000h is added to the address in step S504 and CPU 30 writes the image data from DRAM 37 to this new address. As described above, the write in step S504 causes the image data to be latched in one of latches 60 through 67.

In step S505, CPU 30 determines if all eight addresses in the DRAM table have been written to the unwind logic. If they have not, flow returns to step S503 until all eight addresses have been written to latches 60 to 67 in unwind logic 40.

When a full eight-by-eight bit block of row-ordered image data has been written and latched in latches 60 to 67, flow advances to step S506 in which CPU 30 writes to the eight addresses in the SRAM table set in step S502. This causes the column-ordered image data buffered in buffers 71 through 78 to be written in column-ordered format to the SRAM addresses as described above.

In step S507, the next lower eight-by-eight bit block of image data is selected by adding 1275h to the addresses in the DRAM table and by adding 1h to the addresses shown in the SRAM table. In step S508 CPU 30 determines whether the lower limit of the band of image data has been exceeded. If the lower limit has not been exceeded, flow returns to step S503 in which the newly-selected eight-by-eight bit block of row-ordered image data is reformatted into column-ordered format.

If in step S508 the lower limit of the band of image data has been exceeded, then CPU 30 reinitializes the DRAM table and the SRAM table (step S509). In step S510, CPU 30 selects the next right-most eight-by-eight bit block of image data by adding N times 1h to the addresses shown in the DRAM table and by adding N times 40h to the addresses shown in the SRAM table. (N is the number of times this instruction has been executed and causes the next right-most eight-by-eight bit block of image data to be selected across the band of image data.) In step S511, CPU 30 inspects the resulting addresses to determine if the right-most limit of the band of image data has been exceeded. If it has not been exceeded then flow returns to step S503 to unwind the newly-selected eight-by-eight bit block of row-ordered image data into column-ordered image data into SRAM. On the other hand, if the right-most limit has been exceeded, then the entire band of image has been processed and flow terminates.

The overall process, including a selection of whether or not to unwind image-data, is depicted in the flow chart of Figure 7. The process steps depicted in Figure 7 are stored in EPROM 34 and executed by CPU 30. As shown in step S1, CPU 30 first receives image data and commands over interface 14 and separates the image data from commands. If necessary, in step S2, the image data is decompressed and the decompressed image data is stored in DRAM 37. In step S3, CPU 30 sets or resets the unwind flag in accordance with a command from driver 13 on interface 14. For example, if the data transmitted to DRAM 37 is landscape data, then unwinding is often not needed, and the unwind flag is set low. In step S4, the unwind flag is interrogated. If the unwind flag is set, indicating that data unwinding is desired, then in step S5, CPU 30 retrieves eight bytes of bit map image data from DRAM 37, as detailed in Figure 5, and writes the eight bytes to SRAM (step S6). On the other hand, if in step S4 the unwind flag is not set, CPU 30 retrieves the same eight bytes of bit map image data from DRAM 37, adds D0000h or E0000h to the DRAM address so as to obtain a corresponding SRAM address, and writes them to SRAM directly (step S7).

In step S8, CPU 30 determines whether there is any more data in the image band currently being created in SRAM 42. If there is more data, then flow pro-



ceeds to step S4. If, on the other hand, there is no more data in the image band, then in step S9 CPU 30 initiates DMA transfer of the image data in SRAM 42 to the printer engine. In step S10, PAL 38 switches the SRAM double buffer and flow returns to step S4 so that, in parallel with DMA transfer of image data to the printer engine, processing of the next band of image data may proceed, until in step S11 it is determined that all image data has been processed.

## Claims

1. Apparatus for reformatting image data from a first format into a second format comprising:
  - converting means responsive to image data written to a first address space for converting the image data from the first format to the second format;
  - selecting means for selecting the first format or the second format; and
  - output means responsive to image data written to a second address space for outputting either the image data written to the second address space or the image data converted by said converting means in accordance with the selection by said selecting means.
2. Apparatus according to claim 1, further comprising means for accepting image data over a data interface.
3. Apparatus according to claim 2 wherein at least some of the image data is compressed and further comprising means for decompressing the compressed image data.
4. Apparatus for reformatting image data from a first format into a second format comprising:
  - writing means for writing the image data to a first address space and to a second address space;
  - converting means responsive to said writing means writing image data to the first address space, said converting means for converting the image data from the first format into the second format; and
  - outputting means responsive to said writing means writing to the second address space, said outputting means for outputting data converted by said converting means.
5. Apparatus according to claim 4, further comprising selection means for selecting the first or the second format, and second output means responsive to said writing means writing to the second address space for outputting data in the first format, wherein said output means and said sec-

ond output means are complementarily responsive to said selection means.

6. Apparatus according to either claim 2 or claim 5, further comprising means for accepting commands over an interface, wherein said selecting means is responsive to a command for selecting the first or the second format.
7. Apparatus according to claim 1 or claim 4, wherein the first format is row format and the second format is column format.
8. Apparatus according to claim 4, further comprising an interface, wherein said writing means is adapted to accept image data from said interface.
9. Apparatus according to claim 8, wherein at least some of the image data is compressed and wherein said writing means is adapted to decompress the compressed image data.
10. Apparatus according to claim 9, further comprising a bit map memory, wherein said writing means is adapted to store the decompressed image data in said bit map memory.
11. Apparatus according to claim 4, further comprising a printer and a printer engine, wherein said printer includes a print head adapted to print plural rows simultaneously.
12. A printer comprising:
  - an interface adapted to receive image data and commands;
  - a printer engine including a print head adapted to print plural rows of print information simultaneously; and
  - a controller for controlling said printer engine in accordance with image data and commands received from said interface, said controller including conversion means responsive to image data written to a first address space for converting the image data from a row format into a column format, and output means for outputting the converted image data to said printer engine, said output means being responsive to image data written to a second address space.
13. Apparatus according to claim 12, further comprising selection means for selecting the row format or the column format, wherein said output means is adapted to output the image data in the row format or the converted data in the column format in accordance with said selection means and in response to writing image data to the second address space.

14. Apparatus according to claim 13, further comprising a bit map memory, wherein said controller is adapted to separate commands from image data on said interface and to store image data in said bit map memory. 5
15. Apparatus according to claim 14, wherein at least some of said image data is compressed image data and wherein said controller further comprises decompression means for decompressing the image data. 10
16. Apparatus according to any one of claims 1, 4 or 12, wherein said converting means comprises plural latches for latching respective bytes of image data in the first format and plural buffers, each connected to each of said plural latches, for converting image data into the second format. 15
17. Apparatus according to any one of claims 1, 5 or 12, wherein said output means comprises a DMA channel. 20
18. A printer comprising:  
     an interface adapted to receive row format image data and commands;  
     a bit map memory for storing print information;  
     a printer engine including a print head adapted to print simultaneously plural rows of print information from said bit map memory;  
     a controller operable in accordance with commands from said interface, said controller including plural latches for latching respective bytes of row format image data, and plural buffers, each connected to each of said plural latches, for converting image data latched in said latches into column format image data and for writing the column format image data to said bit map memory. 25
19. Apparatus according to claim 18, wherein said latches are responsive to image data written to a first address space and wherein said buffers are responsive to image data written to said bit map memory. 30
20. Apparatus according to any one of claims 1, 12 or 18, further comprising a first multiplexer for activating one of said plural latches based on address data, and a second multiplexer for activating one of said plural buffers based on address data. 35
21. Apparatus according to claim 20, wherein said first multiplexer is responsive to least significant bits of address data and wherein said second multiplexer is responsive to next least significant 40
22. Apparatus according to claim 18, wherein said controller further comprises selection means responsive to row format image data written to said bit map memory for causing said bit map memory to store the row format image data or the column format image. 45
23. Apparatus according to either claim 13 or claim 22, wherein said selection means is responsive to a command from said interface for selecting the row format or the column format. 50
24. Apparatus according to claim 18, further comprising a second bit map memory, wherein said controller is adapted to separate commands from image data on said interface and to store row format image data in said second bit map memory. 55
25. Apparatus according to claim 24, wherein at least some of said image data is compressed image data and wherein said controller further comprises decompression means for decompressing the image data.
26. Apparatus according to claim 18, further comprising a DMA channel for outputting print information from said bit map memory to said printer engine.
27. Apparatus according to either claim 16 or 18, wherein there are N latches and N buffers, where N is the number of bits in a byte of image data.
28. A printer comprising:  
     an interface adapted to receive image data;  
     a printer engine including a print head adapted to print plural rows of image data simultaneously;  
     a first memory for storing process steps that select a row or a column format, that write image data to a first address space and to a second address space, and that initiate DMA transfer from the second address space to said printer engine;  
     process means for executing the process steps stored in said first memory;  
     converting means responsive to said process means writing to the first address space for converting image data from the first format to the second format;  
     a second memory responsive to said process means writing to the second address space for storing image data;  
     selection means for selecting the first format or the second format in accordance with the process step executed by said process means,

said selection means for causing the second memory to store converted image data from said conversion means or unconverted image data in accordance with the selected format; and

a DMA channel for initiating DMA transfer from said second memory to said printer engine.

29. A printer according to claim 28, further comprising a host computer having a driver for transmitting image data over said interface.

30. A printer according to claim 29, wherein said driver is adapted to transmit commands over said interface, wherein said process means is adapted to distinguish between image data and commands.

31. A printer according to claim 29, wherein said driver is adapted to transmit compressed image data and wherein said first memory stores process steps that decompress the compressed image data.

32. A printer according to any one of claims 17, 26 or 28, wherein said DMA channel includes double buffering.

33. A method for converting image data from a first format to a second format comprising the steps of:

writing first format image data to a first address space;

converting first format image data written to the first address space into second format image data;

writing to a second address space; and  
storing data converted in said converting step in response to writing to the second address space.

34. A method according to claim 33, wherein said writing step writes first format data to the second address space and further comprising the step of selecting whether to store first format data or second format data in said storing step.

35. A method according to claim 33, further comprising the steps of receiving image data over an interface and storing the image data in a bit map memory.

36. A method according to claim 35, wherein at least some of the image data received in said receiving step is compressed image data and further comprising the step of decompressing the compressed image data.

37. A method according to claim 33, further compris-

ing the step of outputting data stored in said storing step over a DMA interface.

38. A method according to claim 37, further comprising the step of printing the data output in said outputting step.

39. A method for printing image data comprising the steps of:

receiving image data and commands;

writing the image data to a first address space;

converting the image data written to the first address space from a first format into a second format;

writing to a second address space;

storing image data converted in said converting step in response to writing to the second address space; and

printing the image data stored in said storing step.

40. A method according to claim 39, wherein said writing step writes image data in the first format data to the second address space and further comprising the step of selecting whether to store first format image data or second format image data in said storing step.

41. A method according to claim 39, further comprising the steps of separating image data from commands and storing the image data in a bit map memory.

42. A method according to claim 41, wherein at least some of the image data is compressed image data and further comprising the step of decompressing the compressed image data.

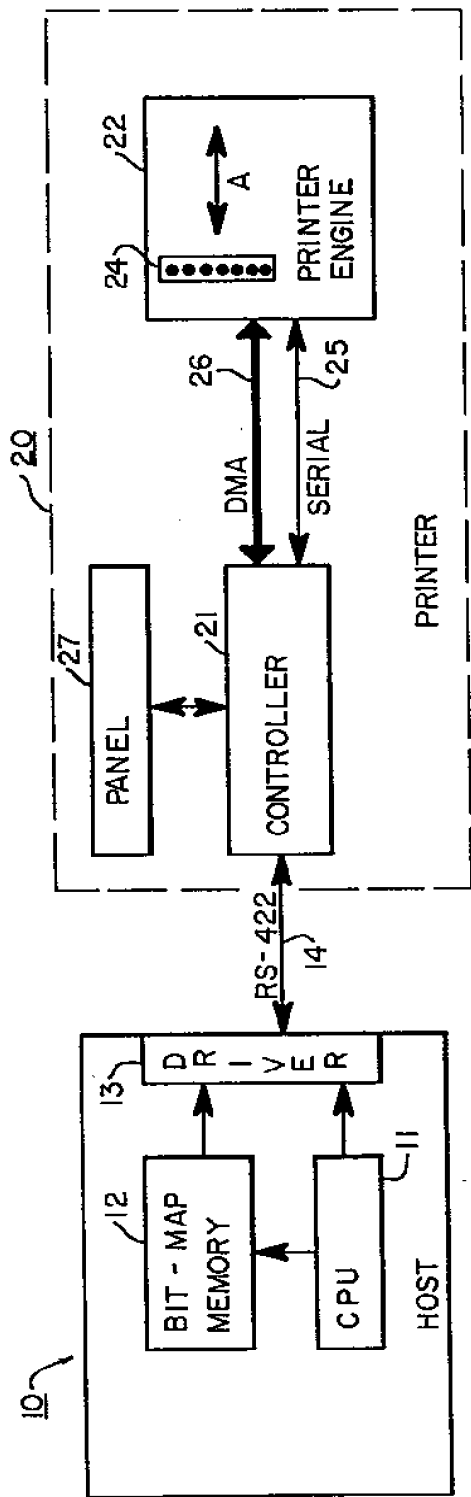
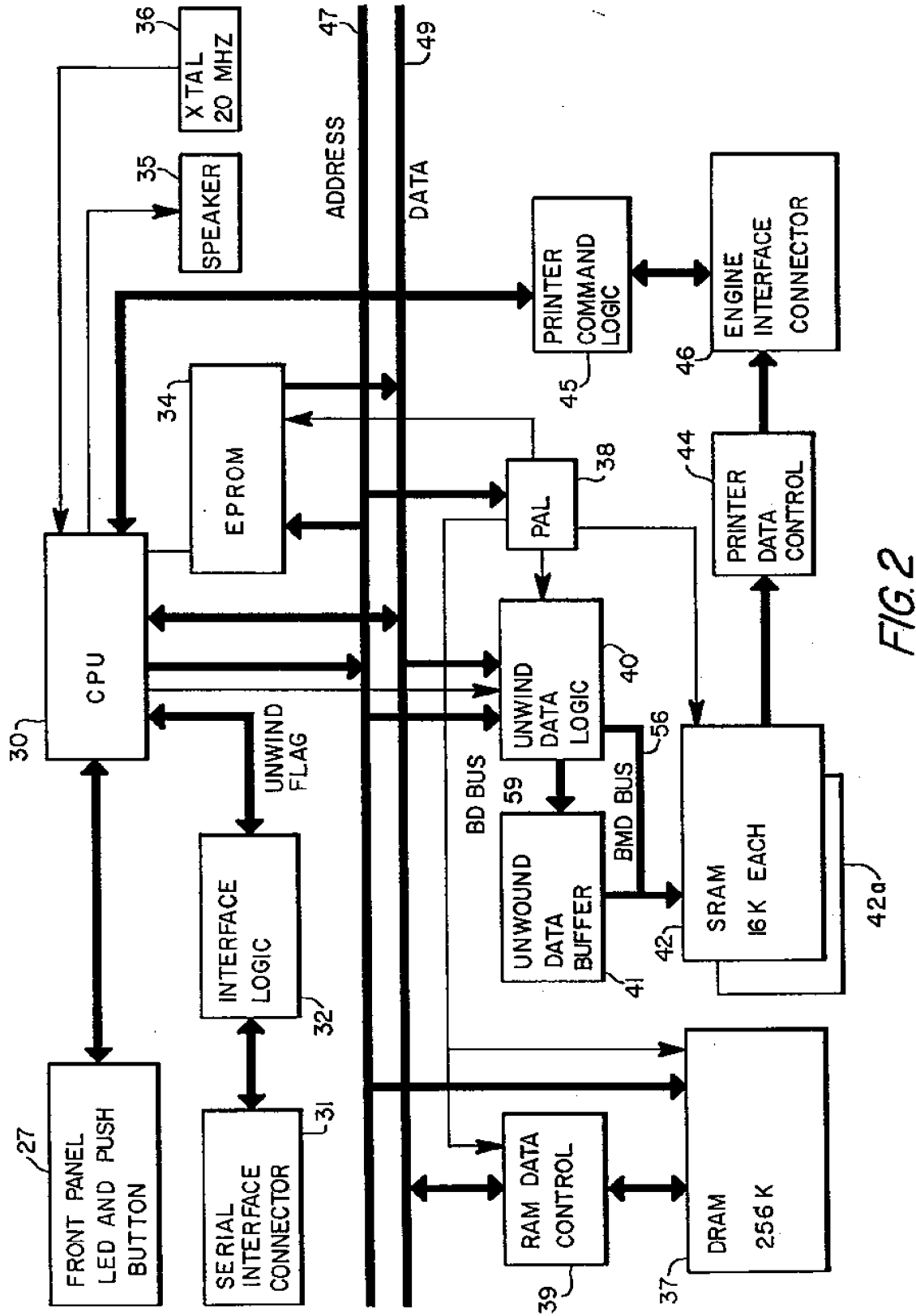


FIG. 1

EPROM	FFFFFh
	F0000h
SRAM 2 (SECOND BUFFER)	EFFFFh
	E0000h
SRAM 1 (FIRST BUFFER)	DFFFFh
	D0000h
UNWIND LOGIC	8FFFFh
	80000h
DRAM	3FFFFFFh
	00000h

FIG. 3



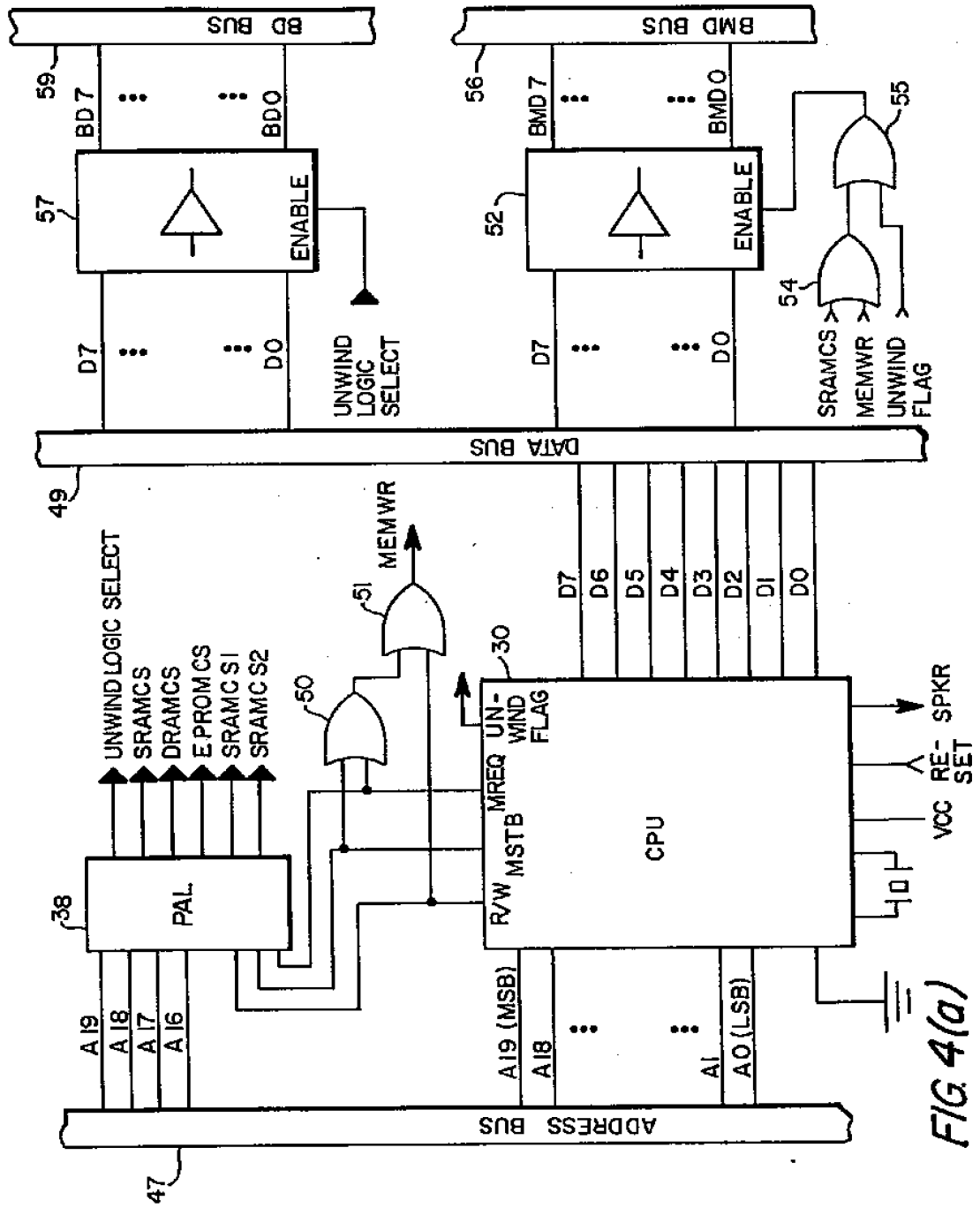
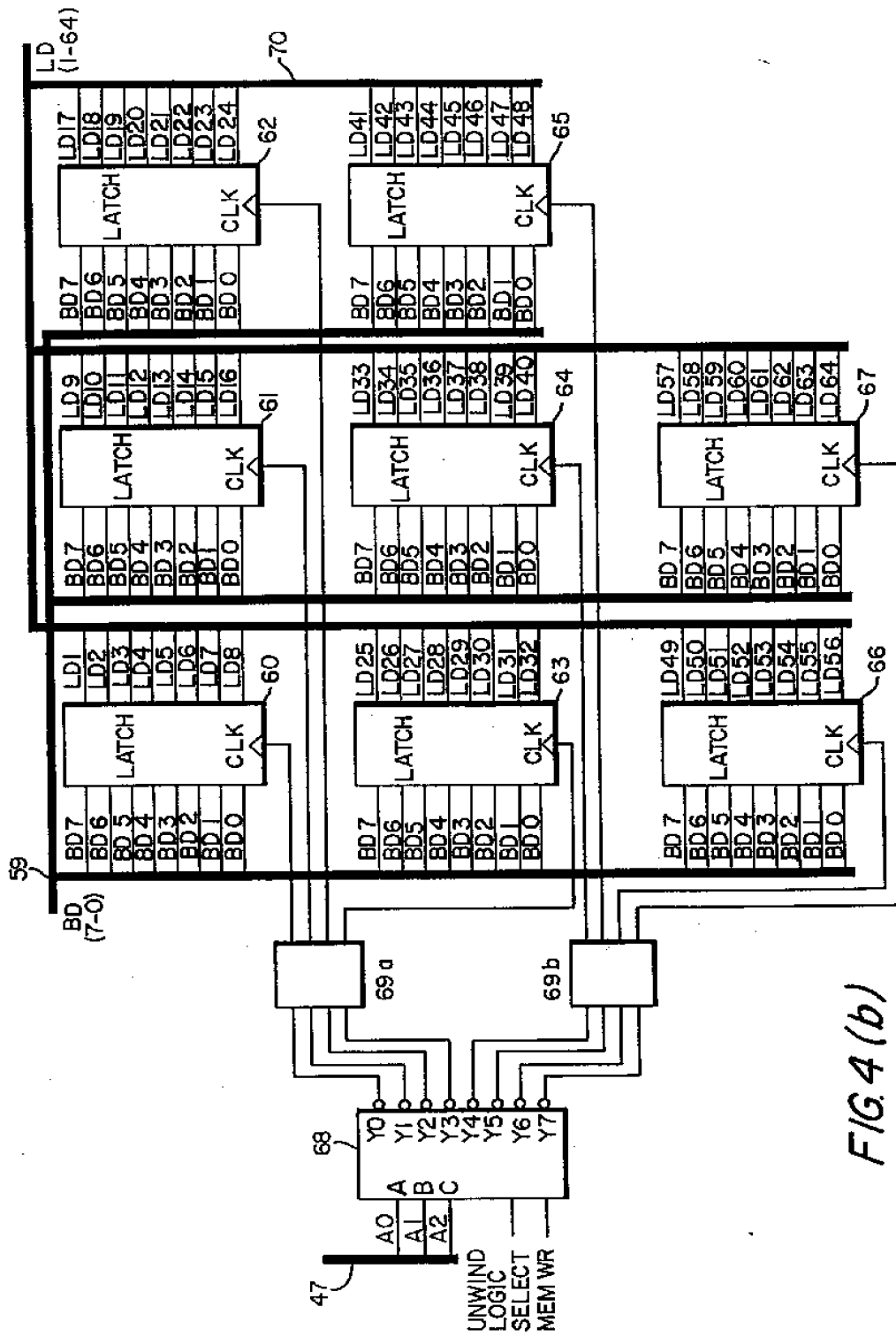


FIG. 4(a)



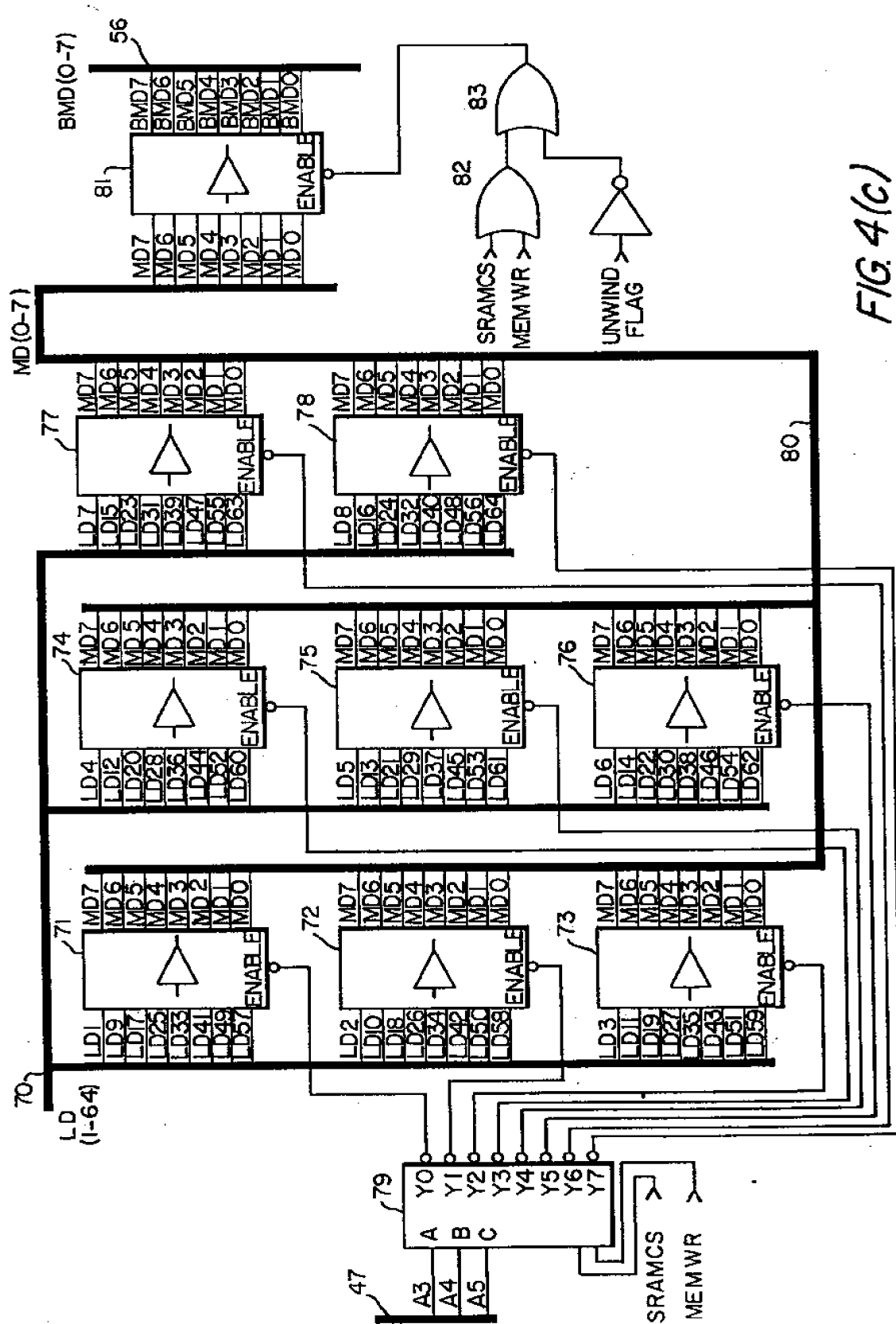
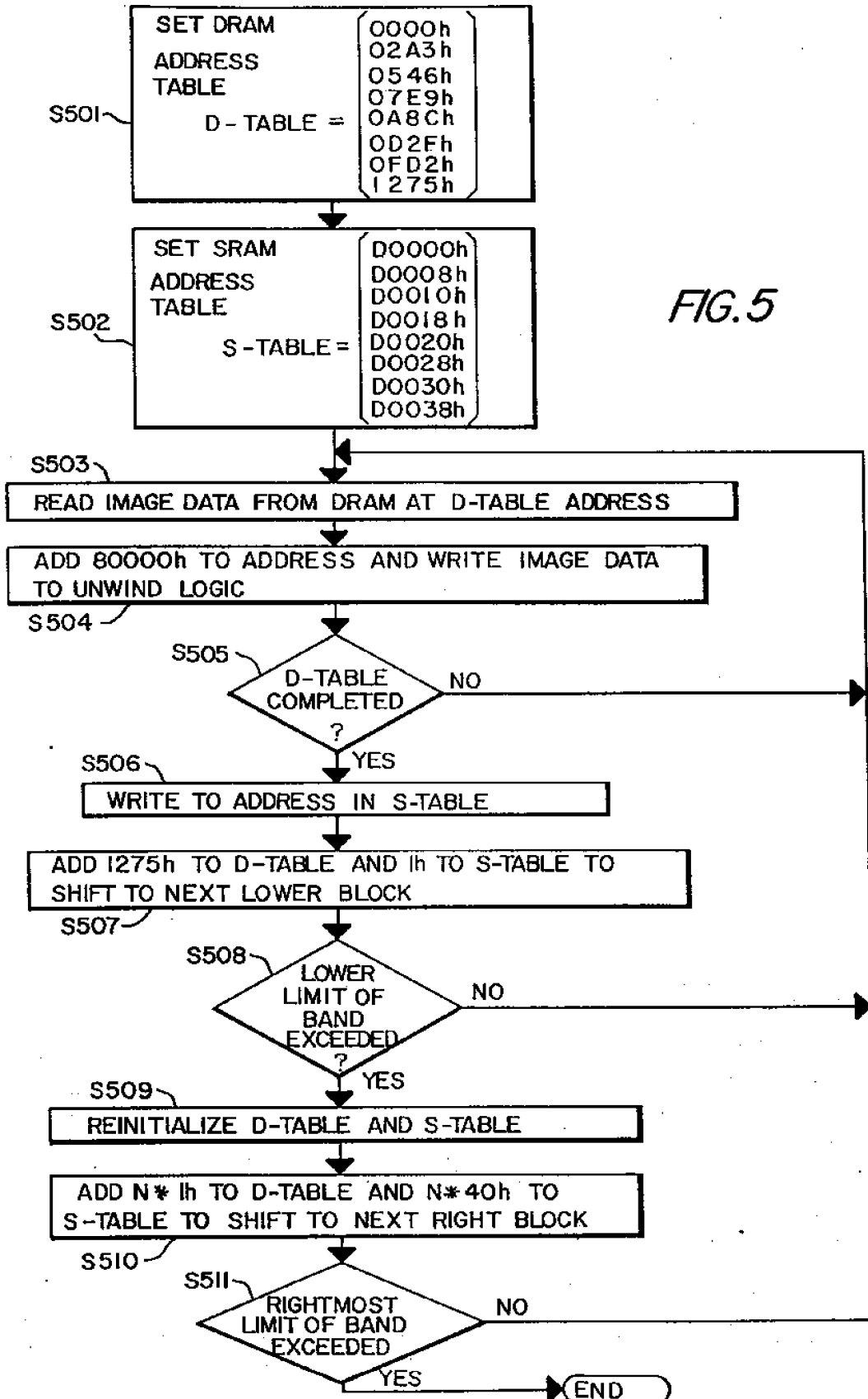


FIG. 4(c)





87	AAO	AA1	AA2	AA3	AA4	AA5	AA6	AA7	AA8	AA9	...	AA5393	AA5394	AA5395	AA5396	AA5397	AA5398	AA5399
	ABO	AB1	AB2	AB3	AB4	AB5	AB6	AB7	AB8	AB9		AB5393	AB5394	AB5395	AB5396	AB5397	AB5398	AB5399
	ACO	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9		AC5393	AC5394	AC5395	AC5396	AC5397	AC5398	AC5399
	ADO	AD1	AD2	AD3	AD4	AD5	AD6	AD7	AD8									AD5399
	AEO	AE1	AE2	AE3	AE4	AE5	AE6	AE7	AE8									AE5399
	AFO	AF1	AF2	AF3	AF4	AF5	AF6	AF7										AF5399
	AGO	AG1	AG2	AG3	AG4	AG5	AG6	AG7										AG5399
	AHO	AH1	AH2	AH3	AH4	AH5	AH6	AH7										AH5399
	BAO	BA1	BA2	BA3	BA4	BA5	BA6	BA7										BA5399
	BBO	BB1	BB2	BB3	BB4	BB5	BB6	BB7										BB5399
	BCO	BC1	BC2	BC3	BC4	BC5	BC6	BC7										BC5399
	BDO	BD1	BD2	BD3	BD4	BD5	BD6	BD7										BD5399
	BEO	BE1	BE2	BE3	BE4	BE5	BE6	BE7										BE5399
	BFO	BF1	BF2	BF3	BF4	BF5	BF6	BF7										BF5399
	BGO	BG1	BG2	BG3	BG4	BG5	BG6	BG7										BG5399
	BHO	BH1	BH2	BH3	BH4	BH5	BH6	BH7										BH5399
	CAO	CA1	CA2	CA3	CA4	CA5												CA5399
	CBO	CB1																CB5399
																		...
																		...
	GGO																	GG5399
	GHO																	GH5399
	HAO																	HA5399
	HBO																	HB5399
	HCO																	HC5399
	HDO																	HD5399
	HEO	HE1																HE5399
	HFO	HF1	HF2												HF5397	HF5398	HF5399	HF5399
	HGO	HG1	HG2	HG3	HG4	HG5	HG6	HG7					HG5394	HG5395	HG5396	HG5397	HG5398	HG5399
	HHO	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	HH9		HH5393	HH5394	HH5395	HH5396	HH5397	HH5398	HH5399

FIG. 6(a)

BYTE ADDRESS IN  
DRAM (HEX)

BYTE CONTENTS

FIG. 6(b)

ROW 1	0	AA0	AA1	AA2	AA3	AA4	AA5	AA6	AA7
	1	AA8	AA9						
	2	AA16							
	...								
ROW 2	2A2	AA5392	AA5393	AA5394	AA5395	AA5396	AA5397	AA5398	AA5399
	2A3	AB0	AB1	AB2	AB3	AB4	AB5	AB6	AB7
	2A4	AB8	AB9						
	...								
ROW 3	545	AB5392							
	546	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
	547	AC8							
	...								
ROW 4	7E8	AC5392							
	7E9	AD0	AD1	AD2	AD3	AD4	AD5	AD6	AD7
	7EA	AD8							
	...								
ROW 5	AB8	AD5392							
	ABC	AE0	AE1	AE2	AE3	AE4	AE5	AE6	AE7
	ABD	AE8							
	...								
ROW 6	D2E	AE5392							
	D2F	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
	D30	AF8							
	...								
ROW 7	FD0	AF5392							
	FD1	AG0	AG1	AG2	AG3	AG4	AG5	AG6	AG7
	FD2	AG8							
	...								
ROW 8	1274	AG5392							
	1275	AH0	AH1	AH2	AH3	AH4	AH5	AH6	AH7
	1276	AH8							
	...								
ROW 9	1517	AH5392							
	1518	BA0	BA1	BA2	BA3	BA4	BA5	BA6	BA7
	1519	BA8							
	...								
ROW 63	A379	HF5392							
	A37A	HG0	HG1	HG2	HG3	HG4	HG5	HG6	HG7
	A37B	HG8							
	...								
ROW 64	A61C	HG5392							
	A61D	HH0	HH1	HH2	HH3	HH4	HH5	HH6	HH7
	A61E	HH8							
	...								
	A8BF	HH5392	HH5393	HH5394	HH5395	HH5396	HH5397	HH5398	HH5399

BYTE ADDRESS IN SRAM (HEX)		BYTE CONTENTS							
COLUMN 1	D0000	AA0	AB0	AC0	AD0	AE0	AF0	AG0	AH0
	D0001	BA0	BB0	BC0	BD0	BE0	BF0	BG0	BH0
	D0002	CA0	CB0	CC0	CD0	CE0	CF0	CG0	CH0
	D0003	DA0	DB0	DC0	DD0	DE0	DF0	DG0	DH0
	D0004	EA0	EB0	EC0	ED0	EE0	EF0	EG0	EH0
	D0005	FA0	FB0	FC0	FD0	FE0	FF0	FG0	FH0
	D0006	GA0	GB0	GC0	GD0	GE0	GF0	GG0	GH0
	D0007	HA0	HB0	HC0	HD0	HE0	HF0	HG0	HH0
COLUMN 2	D0008	AA1	AB1	AC1	AD1	AE1	AF1	AG1	AH1
	D0009	BA1	BB1	BC1	BD1	BE1	BF1	BG1	BH1
	D000A	CA1	CB1	CC1	CD1	CE1	CF1	CG1	CH1
	D000B	DA1	DB1	DC1	DD1	DE1	DF1	DG1	DH1
	D000C	EA1	EB1	EC1	ED1	EE1	EF1	EG1	EH1
	D000D	FA1	FB1	FC1	FD1	FE1	FF1	FG1	FH1
	D000E	GA1	GB1	GC1	GD1	GE1	GF1	GG1	GH1
	D000F	HA1	HB1	HC1	HD1	HE1	HF1	HG1	HH1
COLUMN 3	D0010	AA2	AB2	AC2	AD2	AE2	AF2	AG2	AH2
	D0011	BA2	BB2	BC2	BD2	BE2	BF2	BG2	BH2
	D0012	CA2	CB2	CC2	CD2	CE2	CF2	CG2	CH2
COLUMN 8	D0037	HA6	HB6	HC6	HD6	HE6	HF6	HG6	HH6
	D0038	AA7	AB7	AC7	AD7	AE7	AF7	AG7	AH7
	D0039	BA7	BB7	BC7	BD7	BE7	BF7	BG7	BH7
	D003A	CA7	CB7	CC7	CD7	CE7	CF7	CG7	CH7
	D003B	DA7	DB7	DC7	DD7	DE7	DF7	DG7	DH7
	D003C	EA7	EB7	EC7	ED7	EE7	EF7	EG7	EH7
	D003D	FA7	FB7	FC7	FD7	FE7	FF7	FG7	FH7
	D003E	GA7	GB7	GC7	GD7	GE7	GF7	GG7	GH7
COLUMN 9	D003F	HA7	HB7	HC7	HD7	HE7	HF7	HG7	HH7
	D0040	AA8	AB8	AC8	AD8	AE8	AF8	AG8	AH8
	D0041	BA8	BB8	BC8	BD8	BE8	BF8	BG8	BH8
COLUMN 5400	DA8B7	HA5398	HB5398	HC5398	HD5398	HE5398	HF5398	HG5398	HH5398
	DA8B8	AA5399	AB5399	AC5399	AD5399	AE5399	AF5399	AG5399	AH5399
	DA8B9	BA5399	BB5399	BC5399	BD5399	BE5399	BF5399	BG5399	BH5399
	DA8BA	CA5399	CB5399	CC5399	CD5399	CE5399	CF5399	CG5399	CH5399
	DA8BB	DA5399	DB5399	DC5399	DD5399	DE5399	DF5399	DG5399	DH5399
	DA8BC	EA5399	EB5399	EC5399	ED5399	EE5399	EF5399	EG5399	EH5399
	DA8BD	FA5399	FB5399	FC5399	FD5399	FE5399	FF5399	FG5399	FH5399
	DA8BE	GA5399	GB5399	GC5399	GD5399	GE5399	GF5399	GG5399	GH5399
END	DA8BF	HA5399	HB5399	HC5399	HD5399	HE5399	HF5399	HG5399	HH5399

FIG 6(c)

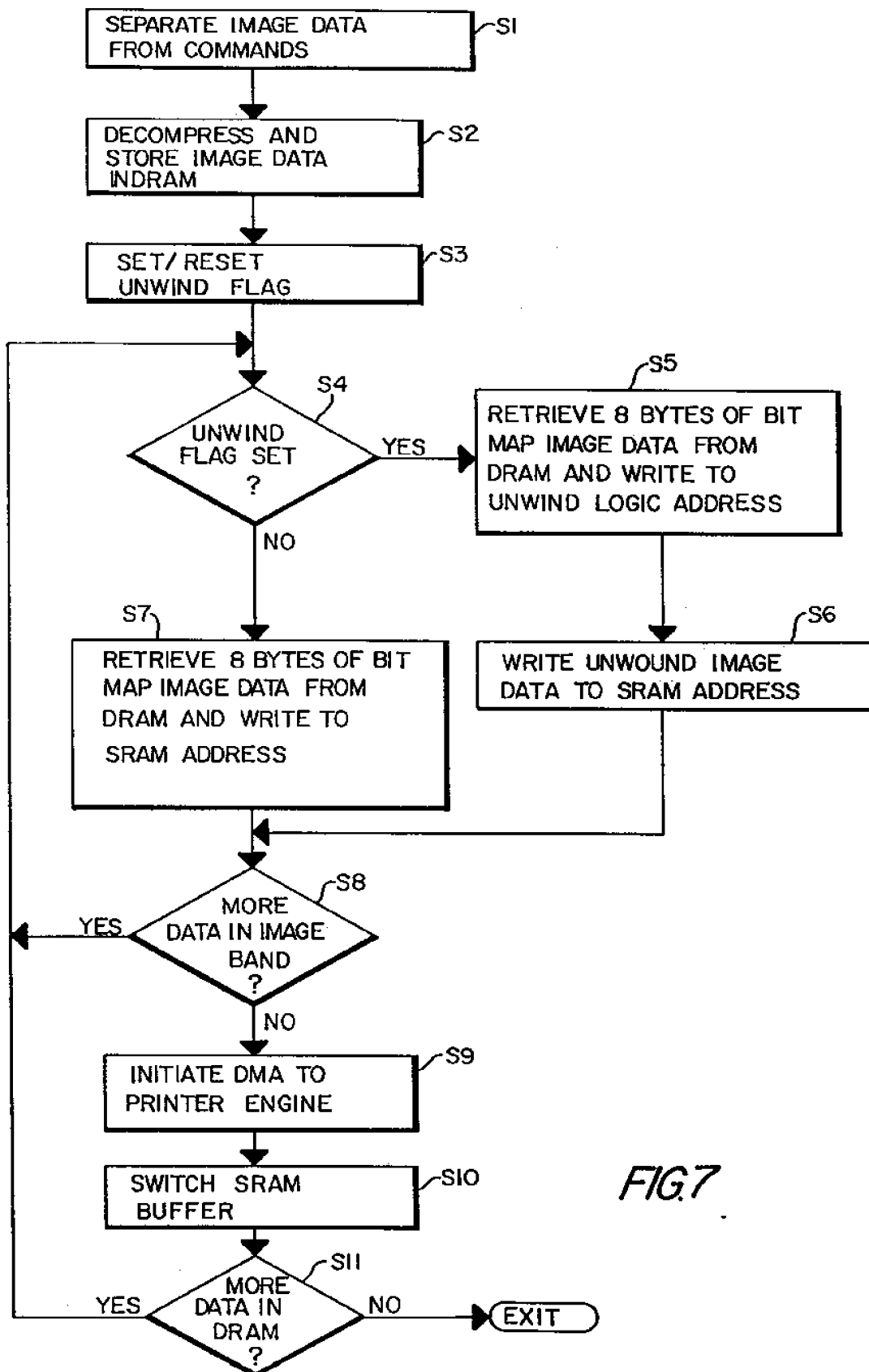


FIG. 7

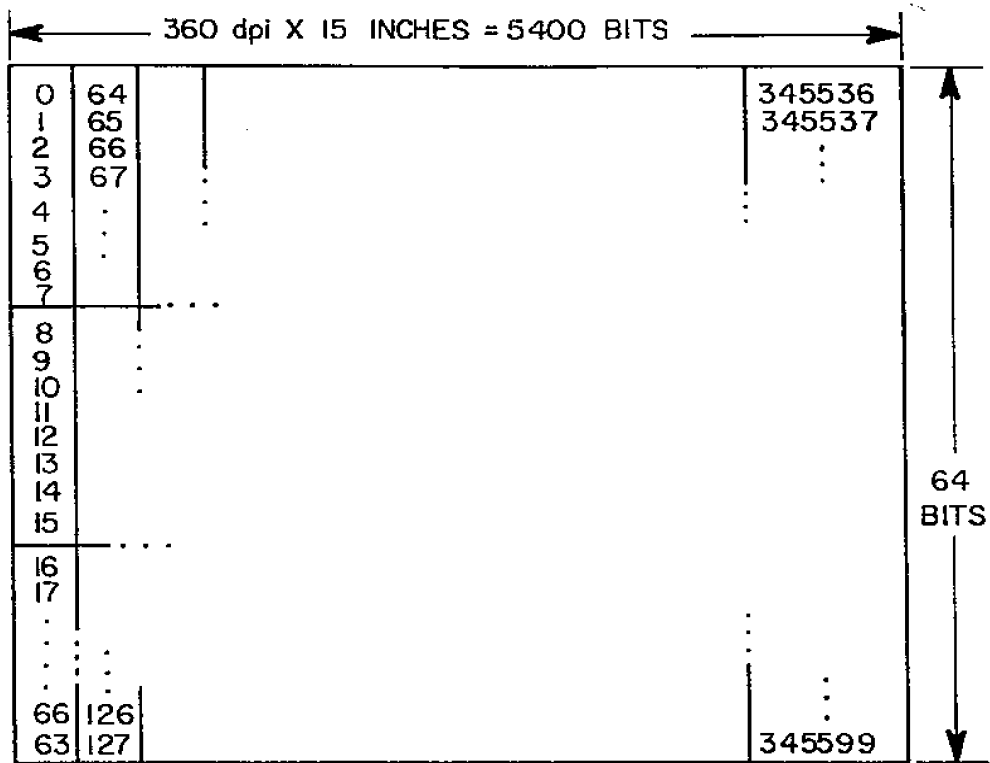


FIG.8(a)

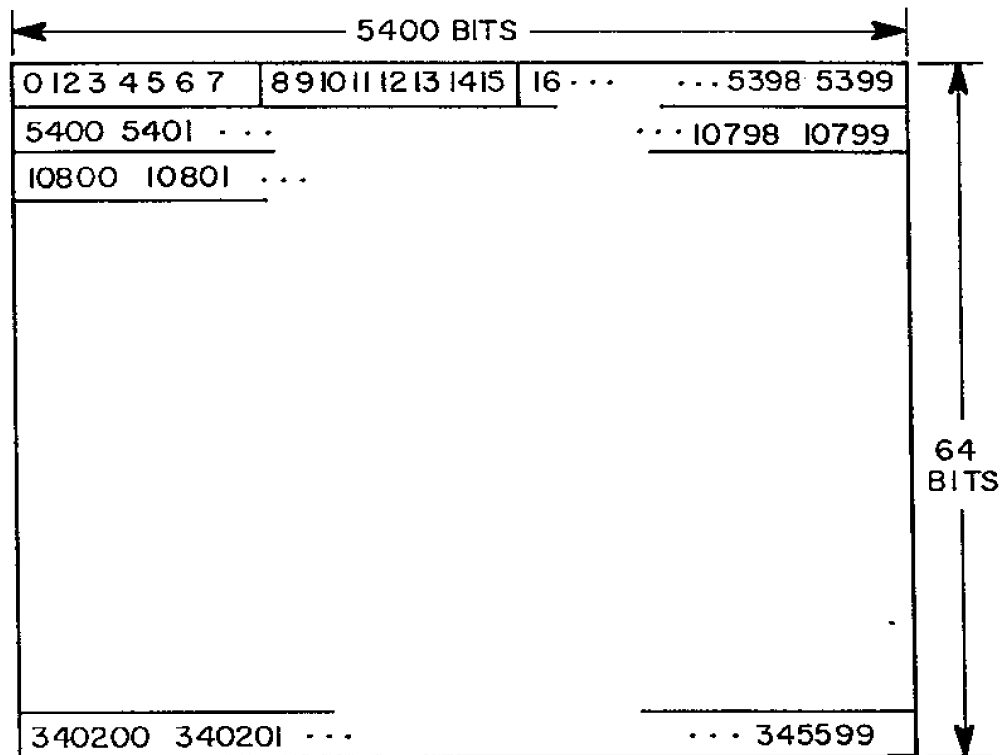


FIG. 8 (b)



European Patent  
Office

## EUROPEAN SEARCH REPORT

Application Number

EP 93 30 4367

DOCUMENTS CONSIDERED TO BE RELEVANT			CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	
Y	US-A-4 691 364 (FUKUZAWA ET AL.)  * the whole document * ----	1,2,4-8, 11-13, 16,18, 19,23, 33,39-41	G06K15/00 G06K15/10
Y	US-A-4 052 699 (MICKA ET AL.)  * the whole document * ----	1,2,4-8, 11-13, 16,18, 19,23, 33,39-41	TECHNICAL FIELDS SEARCHED (Int. Cl.5)  G06K
A	US-A-5 079 739 (PETERSEN) ----		
A	PATENT ABSTRACTS OF JAPAN vol. 11, no. 105 (M-577)3 April 1987 & JP-A-61 252 175 ( OMRON TATEISI ELECTRONICS CO ) 10 November 1986 * abstract * ----		
A	IBM TECHNICAL DISCLOSURE BULLETIN. vol. 33, no. 8, January 1991, NEW YORK US pages 54 - 56 'Simple image compression technique.'  -----		
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 24 SEPTEMBER 1993	Examiner GELEBART Y.C.M.
<p><b>CATEGORY OF CITED DOCUMENTS</b></p> <p>X : particularly relevant if taken alone  Y : particularly relevant if combined with another document of the same category  A : technological background  O : non-written disclosure  P : intermediate document</p> <p>T : theory or principle underlying the invention  E : earlier patent document, but published on, or after the filing date  D : document cited in the application  I : document cited for other reasons  -----  &amp; : member of the same patent family, corresponding document</p>			

EPO FORM 1501 (03.92) (P0401)